IN THE SPECIFICATION:

Please amend the Specification as follows.

Page ~~3~~ 5, paragraph beginning on line 20:

A1

The present invention further includes an apparatus for managing memory in a network switch, wherein the apparatus includes a memory, wherein the memory includes a plurality of memory locations configured to store data therein. A memory address pool having a plurality of available memory addresses arranged therein, wherein each of the plurality of memory addresses corresponds to a specific memory location is provided. A means for managing a memory address pointer, wherein the memory address pointer indicates a next available memory address in the memory address pool is provided, and a means for reading available memory addresses from the memory address pool using a last in first out operation is included. A means for writing released memory addresses into the memory address ~~poolis~~ pool is provided. Finally, a means for adjusting a position of the memory address pointer upon a read or a write operation from the memory address pool is provided, wherein the means for reading and the means for writing are configured to pass off an available memory address when a request to read an address is received during the same clock cycle as a request to write an address.

Page 7, line 32, please add the following new paragraph:

A2

Figure 39 is an illustration of a memory management configuration and method used in one embodiment of the present invention.

Page 16, paragraph beginning on line 9:

A3

The Src Dest Port field of the P-channel message, as stated above, defines the destination and source port addresses, respectively. Each field is 6 bits wide and therefore allows for the addressing of sixty-four ports.

Page 21, paragraph beginning on line 26:

A4

If EPIC modules 20a, 20b, 20c, and GPIC modules 30a and 30b, and CMIC 40 simultaneously request C channel access, then access is granted in round-robin fashion. For a given arbitration time period each of the I/O modules would be provided access to the C channel 81. For example, each GPIC module 30a and 30b would be granted access, followed by the EPIC modules, and finally the CMIC 40. After every arbitration time period the next I/O module with a valid request would be given access to the C channel 81. This pattern would continue as long as each of the I/O modules provides an active C channel 81 access request.

Page 45, paragraph beginning on line 17:

A5

In certain situations, such as a destination lookup failure (DLF)_where a packet is sent to all ports on a VLAN, or a multicast packet, the trunk group bit map table is configured to pickup appropriate port information so that the packet is not sent back to the members of the same source trunk group. This prevents unnecessary traffic on the LAN, and maintains the efficiency at the trunk group.

Page 59, paragraph beginning on line 28 and bridging pages 59 and 60:

A6 The pre-defined algorithm allocates memory locations between the internal memory and the external memory based upon the amount of internal memory available for the egress port of the network switch from which the data packet is to be transmitted by the network switch. When the internal memory available for the egress port from which the data packet is to be transmitted is above a predetermined threshold, then the data packet is stored in the internal memory. ~~Wshen~~ When the internal memory available for the egress port from which the data packet is to be transmitted is below the predetermined threshold value, then the data packet is stored in the external memory.

Page 64, paragraph beginning on line 15:

A7 In the example of the present embodiment, PMMU 70 accepts a new word from an ingress port via CP Bus 100 every 4 clock cycles ~~cycless~~. Further, a new cell is sent to the egress ports every 4 clocks, again via CP Bus 100. Exemplary cases for the maximum and minimum timing for these operations is illustrated in Figure 21.

Page 65, paragraph beginning on line 14:

A8 PMMU 70 further utilizes novel structure and logic within the respective FIFO's to optimally store data within storage units. This structure and logic generally includes generating a glitchless fractional clock pulse from an increment or enable pulse and a clock signal, which is provided to a storage element to enable a data storage operation in a time period in which the data to be stored is most stable. The glitchless fractional clock

A8 pulse, which is generally of a shorter period than the system core clock pulse and asserted high during the same time period or duration that the core clock pulse is asserted high, defines a region in which the data to be stored is in ~~an predicably~~ a predictably stable state. The predictably stable state is a result of the data being stable in the median region of the clock pulse, as opposed to the end regions proximate the rising and falling edge of the clock pulse, where the data tends to be unstable. The structure and logic of the present invention not only allows for storage of data during a predictably stable portion of a clock cycle, but also minimizes overhead consumption via usage of simple space saving elements.

Page 86, paragraph beginning on line 24 and bridging pages 86 and 87:

PPP_ctrl's primary function is to manage the PPP RAM. PPP_ctrl takes write requests from Scheduler 94 and SDU 92 for storage of the FPP and also read requests from CRRU 98 for taking the FPP. However, all requestor's, namely the ESWrite31x0, SDUWrite31x0, & CRRURead31x0 in the present example ~~exanple~~, have at most one bit active at a time. Therefore, all requests are treated independently, and subsequent requests will typically be at most every 4 clocks. PPP_ctrl is also responsible for storing all 32 read/write pointers for the PPP. Each pointer points to the beginning of a packet linked list. SDU 92 and Scheduler 94 blocks will write at an appropriate rate corresponding to the rate at which the acknowledge signals return, and likewise CRRU 98 will read at an equivalent speed. PPP_ctrl will have shallow FIFO's around it to

A9

A9

absorb any transient bursts in requests from other modules. However, in order to meet the speed and bandwidth requirements of SOC 10, PPP_ctrl is ~~confugured~~ configured to service a read request every 8 clock cycles and 2 write requests every 8 clock cycles, all of which are independent events. However, the PPP_ctrl can theoretically operate as fast as the acknowledge signals are received, and therefore, performance of SOC 10 is generally not affected by PPP_ctrl's operation. Further, PPP_ctrl includes three FSM's: first, an Arbiter_FSM that is responsible for placing requests into the Command Queue; second, an Address_FSM that is responsible for the issuance of addresses and control signals to CBP 50; and third, a Data_FSM that is responsible for moving data, selecting multiplexers that go to the wr_data input, and asserting all the correct Increment and decrement pulses.

Page 89, paragraph beginning on line 12:

A10

With regard to timing issues, SFAP 97 will activate its request signal during the same clock that SFAP 97 makes a ~~comand~~ command valid, as shown in Figure 22. Eventually, the SDRAM Controller 96 will reply with an Ack signal. This may be a few hundred clock cycles, if Controller 96 is dealing with a priority situation with the SAU's or other modules. While waiting for the Ack signal, the SFAP 97 can continue to access its internal SRAM. Once the Ack signal is returned to SFAP 97, it needs to switch off its internal SRAM accesses. In a minimum of two clocks after Ack goes active, Xfr will go active initiating a read or write transfer of eight words with the SFAP 97 internal SRAM.

A10

Due to clock speed differences between the SOC 10 and SDRAM, there may be one or more idle clock cycles during this transfer. During any idle clocks, transfer will be inactive, as Ack remains active.

Page 93, paragraph beginning on line 14:

A11

When multiplexing data for SDRAM, attention is needed to make sure the cell data bytes are sent in the correct order. Specifically, for ~~examplr~~ example, the sideband bytes are sent first (coming from bits 256-292 of the first SAU word), followed by data bytes 15-0 (bits 127-0 of the same word), then data bytes 31-16 (bits 255-128). SDRAM to SDU bus / SDU format. The bus is 128 bits wide, per the SDRAM interface, and time multiplexed. Data will be latched inside SDU 92 to put together 312-bit words. The bus is basically in SDRAM format, except the NC Header field replaces Slot Size and Copy Count. A "Release" flag is added to inform the SDU logic that this slot should be released back to the free-slot pointer pool. This is generally set for slots not containing multicast or broadcast cells.